

LENGUAJES DE PROGRAMACIÓN

Solución al examen de Septiembre 2026

PREGUNTA 1 (3 puntos)

Indique la veracidad o falsedad de cada una de las afirmaciones siguientes, **explicando detalladamente** en cada caso el motivo de su respuesta.

- A. (0.5 puntos) Java fue el primer lenguaje en introducir los conceptos de puntero y de gestión dinámica de la memoria.
- B. (0.5 puntos) El lenguaje Ada proporciona facilidades para el tratamiento de las excepciones.
- C. (0.5 puntos) En un bucle lógico postcondición se ejecuta siempre el cuerpo del bucle al menos una vez.
- D. (0.5 puntos) En el lenguaje C++, si en el programa no se inicializa una variable local de un tipo básico, la implementación lo hace a su valor por defecto.
- E. (0.5 puntos) El algoritmo de la STL

`remove_copy(pBegin, pEnd, pResultBegin, val)`

copia la secuencia origen en la secuencia destino, eliminando de la copia los elementos cuyo valor sea menor que val.

- F. (0.5 puntos) Una expresión lambda en C++, es una técnica de C++ que permite definir una función anónima que será creada en tiempo de ejecución.

Solución a la Pregunta 1

- A** Falso Véase la página 45 del texto base.
- B** Verdadero Véase la página 48 del texto base.
- C** Verdadero Véase la página 263 del texto base.
- D** Falso Véanse las páginas 134 y 135 del texto base.
- E** Falso Véase la página 556 del texto base.
- F** Verdadero Véase la página 568 del texto base.

PREGUNTA 2 (1 punto)

Escriba la salida por consola producida al ejecutar el programa en C++ escrito a continuación.

```
#include <iostream>
#include <iomanip>

double a = 5, b = 10, c = 1;

int main() {
    double v[4] = { 1.5, 2.5, 3.5, 4.5 };
    double *p = v;
    b = *(p + 2);
    std::cout << std::fixed << std::setprecision(2);
    std::cout << v[2] << std::endl;
    std::cout << *p << std::endl;
    std::cout << b << std::endl;
    {
        double b = 20;
        double c = 30;
        v[2] = b;
        a += c;
    }
    p += 2;
    std::cout << *p << std::endl;
    std::cout << std::setprecision(1) << v[2] << std::endl;
    std::cout << a << std::endl;
    std::cout << b << std::endl;
    std::cout << c << std::endl;
    std::cout << p[0] << std::endl;
    return 0;
}
```

Solución a la Pregunta 2

La salida por consola producida al ejecutar el programa es:

```
3.50
1.50
3.50
20.00
20.0
35.0
3.5
1.0
20.0
```

PREGUNTA 3 (1 punto)

Escriba la salida por consola producida al ejecutar el programa en C++ escrito a continuación.

```
#include <iostream>
#include <map>
#include <string>

int main() {
    std::map<std::string, int> notas;
    notas["A"] = 7;
    notas["B"] = 5;
    notas["C"] = 9;
    notas["D"] = 4;
    notas["E"] = 10;

    std::cout << notas.count("A") << std::endl;
    std::cout << notas.count("AE") << std::endl;

    std::map<std::string, int>::iterator p = notas.find("C");
    while ( p != notas.end() ) {
        std::cout << p->first << " "
                  << p->second << std::endl;
        p++;
    }
    notas["B"] = notas["B"] + 2;
    notas.erase("D");
    p = notas.begin();
    while ( p != notas.end() ) {
        std::cout << p->first << " "
                  << p->second << std::endl;
        p++;
    }
    return 0;
}
```

Solución a la Pregunta 3

La salida por consola producida al ejecutar el programa es:

```
1
0
C 9
D 4
E 10
A 7
B 7
C 9
E 10
```

PREGUNTA 4 (2.5 puntos)

En una empresa de fabricación de radiadores se desea repartir 20000 euros entre los vendedores para premiar su productividad.

En un fichero de texto llamado `vendedores.txt` se encuentra la siguiente información en tres columnas, separadas por un espacio: nombre y apellidos del vendedor, el importe de las ventas que ha realizado y el número de nuevos clientes que ha captado durante ese año. Los nombres y apellidos de los vendedores aparecen unidos mediante el carácter "_" para formar un único campo sin espacios.

A continuación, se muestra un ejemplo del contenido del fichero.

```
Juan_Perez_Garcia 397820 25
Maria_Lopez_Sanchez 420047.5 31
Pedro_Martin_Ruiz 380800.1 22
Ana_Gomez_Perez 510750.23 28
Laura_Fernandez_Diaz 295465.4 19
Carlos_Jimenez_Soto 470843.2 35
Elena_Ruiz_Moreno 367864.03 27
```

El importe del incentivo por productividad se calcula de la forma siguiente:

$$P_a = R \cdot \left(0.5 \cdot \frac{V_a}{\sum_{i=1}^N V_i} + 0.5 \cdot \frac{C_a}{\sum_{i=1}^N C_i} \right)$$

donde

P_a es el incentivo económico asignado al vendedor a .

R es el importe total para repartir (en este caso, 20000 euros).

N es el número total de vendedores.

V_a es el importe de las ventas del vendedor a .

$\sum_{i=1}^N V_i$ es el importe total de las ventas realizadas por todos los vendedores.

C_a es el número de nuevos clientes captados por el vendedor a .

$\sum_{i=1}^N C_i$ es el número total de nuevos clientes que han captado todos los vendedores.

Escriba un programa en C++ que realice las tareas siguientes:

1. Abrir y leer el contenido del fichero, almacenando cada uno de los tres campos (nombre y apellidos, importe de ventas y número de nuevos clientes) en un vector.

Si se produce error, mostrar un mensaje en la consola indicándolo y terminar el programa.

2. Escribir en la consola una fila por cada vendedor, en 4 columnas:

- El nombre y apellidos del vendedor, con una anchura fija de 25 caracteres.
- El importe de sus ventas, con una anchura fija de 15 caracteres.
- El número de sus nuevos clientes, con una anchura fija de 10 caracteres.
- El importe de su incentivo por productividad, con una anchura fija de 15 caracteres.

El programa debe escribir los valores de los importes de las ventas y los incentivos por productividad con 2 dígitos decimales.

Solución a la Pregunta 4

```

1 #include <iostream>
2 #include <fstream>
3 #include <vector>
4 #include <string>
5 #include <iomanip>
6
7 const double IMPORTE_TOTAL = 20000.0;
8
9 int main() {
10     std::ifstream archivo("vendedores.txt", std::ios::in);
11     if( !archivo ) {
12         std::cerr << "Error al abrir el fichero\n";
13         return 1;
14     }
15     std::vector<std::string> nombres;
16     std::vector<double> ventas;
17     std::vector<int> clientes;
18     std::string nombre;
19     double venta;
20     int cliente;
21     while ( archivo >> nombre >> venta >> cliente) {
22         nombres.push_back(nombre);
23         ventas.push_back(venta);
24         clientes.push_back(cliente);
25     }
26     archivo.close();
27     double totalVentas = 0.0;
28     int totalClientes = 0;
29     for(unsigned i=0; i<ventas.size(); i++) {
30         totalVentas += ventas[i];
31         totalClientes += clientes[i];
32     }
33     std::cout << std::fixed << std::setprecision(2)
34         << std::setw(25) << "Nombre"
35         << std::setw(15) << "Ventas"
36         << std::setw(10) << "Clientes"
37         << std::setw(15) << "Incentivo"
38         << std::endl;
39     for(unsigned i=0; i<nombres.size(); i++) {
40         double pesoVentas = ventas[i] / totalVentas;
41         double pesoClientes = double(clientes[i]) / totalClientes;
42         double incentivo = IMPORTE_TOTAL *
43             ( 0.5*pesoVentas + 0.5*pesoClientes );
44         std::cout << std::setw(25) << nombres[i]
45             << std::setw(15) << ventas[i]
46             << std::setw(10) << clientes[i]
47             << std::setw(15) << incentivo << std::endl;
48     }
49     return 0;
50 }

```

PREGUNTA 5 (2.5 puntos)

La evaluación de los alumnos de una asignatura del Grado en Matemáticas se realiza mediante dos pruebas obligatorias: un trabajo y un examen.

Cada una de las pruebas se puntúa entre 0 y 10 puntos.

Si se obtiene una nota igual o superior a 5 puntos en una prueba, dicha prueba se considera aprobada. En caso contrario, se considera suspensa.

Para aprobar la asignatura es necesario aprobar ambas pruebas. El criterio de evaluación es el siguiente:

- Si el alumno ha realizado una de las pruebas pero no la otra, su nota en la asignatura es cero.
- Si el alumno ha realizado ambas pruebas y ha suspendido al menos una de ellas, su nota en la asignatura será la menor de las dos notas obtenidas.
- Si el alumno ha aprobado ambas pruebas, su nota en la asignatura es la media aritmética de las notas obtenidas en ambas pruebas.

Las notas obtenidas por los alumnos en el examen y el trabajo se almacenan en dos *ficheros de calificaciones* llamados *notasExamen.txt* y *notasTrabajo.txt* respectivamente. Los *ficheros de calificaciones* son ficheros de texto en los que cada línea contiene el identificador del alumno (dato de tipo string) y su nota (dato de tipo double), separados por uno o más espacios en blanco.

En cada fichero solo aparecen los alumnos que han realizado la prueba correspondiente, por lo que puede darse el caso de que un alumno aparezca en uno de los ficheros y no en el otro. Aquellos alumnos que han realizado ambas pruebas aparecen en ambos ficheros.

Puede suponer que no existen identificadores repetidos dentro de un mismo fichero. También, que los ficheros no tienen errores de formato.

Escriba el código C++ indicado a continuación, empleando la estructura siguiente:

```
struct NotaAlumno {  
    double notaExamen;  
    double notaTrabajo;  
    double notaAsignatura;  
};
```

5.1 (0.75 puntos) Escriba una función en C++ con el prototipo siguiente:

```
std::map<std::string, double>
    leeFichNotas(std::string nombreFich)
        throw (std::invalid_argument);
```

que lea el contenido del *fichero de calificaciones* cuyo nombre es pasado como argumento y que devuelva un mapa, donde la clave es el identificador del alumno y el valor es su nota en la prueba.

Si se produce un error al abrir el fichero, la función debe lanzar una excepción.

5.2 (0.75 puntos) Escriba una función en C++ con el prototipo siguiente:

```
std::map<std::string, NotaAlumno>
    calculaNotasFinales(
        const std::map<std::string, double> &mNotasTrabajo,
        const std::map<std::string, double> &mNotasExamen);
```

tal que el primer y segundo argumentos son referencias a los mapas con las calificaciones de los alumnos evaluados en el trabajo y el examen, respectivamente.

La función debe devolver un mapa cuya clave sea el identificador del alumno y cuyo valor sea una estructura `NotaAlumno` que almacena sus calificaciones en las pruebas y en la asignatura.

Si un alumno no ha realizado una prueba, la nota correspondiente se almacenará con valor -1 . Como las notas válidas están comprendidas entre 0 y 10, el valor -1 se utilizará para representar que una prueba no ha sido realizada por el alumno.

En el mapa devuelto por la función deben aparecer todos los alumnos que aparecen al menos en uno de los mapas pasados como argumento.

5.3 (1 punto) Escriba un programa principal en C++ que realice las acciones siguientes:

1. Invocando las funciones definidas anteriormente, obtener un mapa cuya clave sea el identificador del alumno y cuyo valor sea la estructura `NotaAlumno` que almacena las calificaciones del alumno en las pruebas y en la asignatura.
Si se produce un error al abrir alguno de los *ficheros de calificaciones*, mostrar un mensaje en la consola indicándolo y terminar.
2. Mediante un mensaje en la consola, solicitar al usuario que introduzca por teclado el identificador del alumno cuyas notas desea consultar.

3. Leer el valor introducido por el usuario, almacenándolo en una variable llamada `alumnoID`.
4. Si no existe ningún elemento en el mapa cuya clave coincida con el identificador introducido por el usuario, escribir en la consola “Alumno no encontrado” y terminar.
5. Escribir en la consola la nota en las pruebas que haya realizado y en la asignatura, con dos dígitos decimales, del alumno cuyo identificador ha introducido el usuario.

Si el alumno no ha realizado una prueba, en el lugar de la calificación en esa prueba se debe escribir NP (No Presentado). Por ejemplo:

Trabajo: 8.50

Examen: NP

Asignatura: 0.00

6. Terminar.

Solución a la Pregunta 5

```

1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 #include <map>
5 #include <iomanip>
6 #include <stdexcept>
7 #include <algorithm>
8
9 // Valor -1: prueba no realizada
10 struct NotaAlumno {
11     double notaExamen;
12     double notaTrabajo;
13     double notaAsignatura;
14 };
15
16 const int CODIGO_NP = -1;
17
18 // 5.1
19 std::map<std::string, double> leeFichNotas(std::string nombreFich)
20     throw (std::invalid_argument) {
21     // Abre el fichero
22     std::ifstream fich(nombreFich.c_str());
23     if (!fich.is_open())
24         throw std::invalid_argument(nombreFich);
25     // Escribe el contenido del fichero en el mapa
26     std::map<std::string, double> notas;
27     std::string idAlumno;
28     double nota;
29     while (fich >> idAlumno >> nota)
30         notas[idAlumno] = nota;
31     return notas;
32 }
33
34 // Funcion auxiliar
35 double calculaNotaAsignatura(double notaTrabajo,
36                             double notaExamen) {
37     // Ha realizado solo una de las pruebas
38     if (notaTrabajo == CODIGO_NP || notaExamen == CODIGO_NP)
39         return 0.0;
40     // Ha suspendido alguna prueba
41     if (notaTrabajo < 5.0 || notaExamen < 5.0)
42         return std::min(notaTrabajo, notaExamen);
43     // Ha aprobado ambas pruebas
44     return (notaTrabajo + notaExamen) / 2.0;
45 }
46
47 // 5.2
48 std::map<std::string, NotaAlumno> calculaNotasFinales(
49     const std::map<std::string, double> &mNotasTrabajo,
50     const std::map<std::string, double> &mNotasExamen) {
51     std::map<std::string, NotaAlumno> resultado;

```

```

52 // Procesar alumnos que han hecho el trabajo
53 for (std::map<std::string, double>::const_iterator it =
54     mNotasTrabajo.begin(); it != mNotasTrabajo.end(); ++it) {
55     NotaAlumno na;
56     na.notaTrabajo = it->second;
57     std::map<std::string, double>::const_iterator itExamen =
58         mNotasExamen.find(it->first);
59     if (itExamen != mNotasExamen.end())
60         na.notaExamen = itExamen->second;
61     else
62         na.notaExamen = CODIGO_NP;
63     na.notaAsignatura =
64         calculaNotaAsignatura(na.notaTrabajo, na.notaExamen);
65     resultado[it->first] = na;
66 }
67 // Procesar alumnos que han hecho solo el examen
68 for (std::map<std::string, double>::const_iterator it =
69     mNotasExamen.begin(); it != mNotasExamen.end(); ++it)
70     if (resultado.find(it->first) == resultado.end()) {
71         NotaAlumno na;
72         na.notaTrabajo = CODIGO_NP;
73         na.notaExamen = it->second;
74         na.notaAsignatura =
75             calculaNotaAsignatura(na.notaTrabajo, na.notaExamen);
76         resultado[it->first] = na;
77     }
78 return resultado;
79 }
80
81 // 5.3
82 int main() {
83     try {
84         // 5.3.1
85         std::map<std::string, double> mNotasTrabajo =
86             leeFichNotas("notasTrabajo.txt");
87         std::map<std::string, double> mNotasExamen =
88             leeFichNotas("notasExamen.txt");
89         std::map<std::string, NotaAlumno> mAlumnos =
90             calculaNotasFinales(mNotasTrabajo, mNotasExamen);
91         // 5.3.2
92         std::cout << "Identificador del alumno: ";
93         // 5.3.3
94         std::string alumnoID;
95         std::cin >> alumnoID;
96         // 5.3.4
97         std::map<std::string, NotaAlumno>::iterator it =
98             mAlumnos.find(alumnoID);
99         if (it == mAlumnos.end()) {
100             std::cout << "Alumno no encontrado\n";
101             return 0;
102         }
103         // 5.3.5
104         std::cout << std::fixed << std::setprecision(2) << "Trabajo: ";
105         if (it->second.notaTrabajo == CODIGO_NP)

```

```
106         std::cout << "NP";
107     else
108         std::cout << it->second.notaTrabajo;
109     std::cout << "\nExamen: ";
110     if (it->second.notaExamen == CODIGO_NP)
111         std::cout << "NP";
112     else
113         std::cout << it->second.notaExamen;
114     std::cout << "\nAsignatura: " << it->second.notaAsignatura << std::endl;
115 } catch (std::invalid_argument &e) { // 5.3.1
116     std::cerr << "Error al abrir el fichero " << e.what() << std::endl;
117     return 1;
118 }
119 // 5.3.6
120 return 0;
121 }
```